

Optimal searching of gapped repeats in a word

Maxime Crochemore* Roman Kolpakov[†]
 Gregory Kucherov[‡]

Abstract

Following (Kolpakov et al., 2013; Gawrychowski and Manea, 2015), we continue the study of α -gapped repeats in strings, defined as factors uvu with $|uv| \leq \alpha|u|$. Our main result is the $O(\alpha n)$ bound on the number of *maximal* α -gapped repeats in a string of length n , previously proved to be $O(\alpha^2 n)$ in (Kolpakov et al., 2013). For a closely related notion of maximal δ -subrepetition (maximal factors of exponent between $1 + \delta$ and 2), our result implies the $O(n/\delta)$ bound on their number, which improves the bound of (Kolpakov et al., 2010) by a $\log n$ factor.

We also prove an algorithmic time bound $O(\alpha n + S)$ (S size of the output) for computing all maximal α -gapped repeats. Our solution, inspired by (Gawrychowski and Manea, 2015), is different from the recently published proof by (Tanimura et al., 2015) of the same bound. Together with our bound on S , this implies an $O(\alpha n)$ -time algorithm for computing all maximal α -gapped repeats.

1 Introduction

Notation and basic definitions. Let $w = w[1]w[2] \dots w[n] = w[1..n]$ be an arbitrary word. The length n of w is denoted by $|w|$. For any $1 \leq i \leq j \leq$

*King's College London, London WC2R 2LS, UK and Université Paris-Est, France, Maxime.Crochemore@kcl.ac.uk

[†]Lomonosov Moscow State University, Leninskie Gory, Moscow, 119992 Russia, foroman@mail.ru

[‡]LIGM, Université Paris-Est Marne-la-Vallée, 77454 Marne-la-Vallée CEDEX 2, France, Gregory.Kucherov@univ-mlv.fr

n , word $w[i] \dots w[j]$ is called a *factor* of w and is denoted by $w[i \dots j]$. Note that notation $w[i \dots j]$ denotes two entities: a word and its occurrence starting at position i in w . To underline the second meaning, we will sometimes use the term *segment*. Speaking about the equality between factors can also be ambiguous, as it may mean that the factors are identical words or identical segments. If two factors u, v are identical words, we call them *equal* and denote this by $u = v$. To express that u and v are the same segment, we use the notation $u \equiv v$. For any $i = 1 \dots n$, factor $w[1 \dots i]$ (resp. $w[i \dots n]$) is a *prefix* (resp. *suffix*) of w . By *positions* on w we mean indices $1, 2, \dots, n$ of letters in w . For any factor $v \equiv w[i \dots j]$ of w , positions i and j are called respectively *start position* and *end position* of v and denoted by $beg(v)$ and $end(v)$ respectively. Let u, v be two factors of w . Factor u is *contained* in v iff $beg(v) \leq beg(u)$ and $end(u) \leq end(v)$. Letter $w[i]$ is *contained* in v iff $beg(v) \leq i \leq end(v)$.

A positive integer p is called a *period* of w if $w[i] = w[i + p]$ for each $i = 1, \dots, n - p$. We denote by $per(w)$ the *smallest period* of w and define the *exponent* of w as $exp(w) = |w|/per(w)$. A word is called *periodic* if its exponent is at least 2. Occurrences of periodic words are called *repetitions*.

Repetitions, squares, runs. Patterns in strings formed by repeated factors are of primary importance in word combinatorics [22] as well as in various applications such as string matching algorithms [12, 9], molecular biology [14], or text compression [24]. The simplest and best known example of such patterns is a factor of the form uu , where u is a nonempty word. Such repetitions are called *squares*. Squares have been extensively studied. While the number of all square occurrences can be quadratic (consider word a^n), it is known that the number of *primitively-rooted* squares is $O(n \log n)$ [9], where a square uu is *primitively-rooted* if the exponent of u is not an integer greater than 1. An optimal $O(n \log n)$ -time algorithm for finding all primitively-rooted squares was proposed in [5].

Repetitions can be seen as a natural generalization of squares. A repetition in a given word is called *maximal* if it cannot be extended by at least one letter to the left nor to the right without changing (increasing) its minimal period. More precisely, a repetition $r \equiv w[i \dots j]$ in w is called *maximal* if it satisfies the following conditions:

1. $w[i - 1] \neq w[i - 1 + per(r)]$ if $i > 1$,
2. $w[j + 1 - per(r)] \neq w[j + 1]$ if $j < n$.

For example, word `cababaaa` has two maximal repetitions: `ababa` and `aaa`. Maximal repetitions are usually called *runs* in the literature. Since any repetition is contained in some run, the set of all runs can be considered as a compact encoding of all repetitions in the word. This set has many useful applications, see, e.g., [7]. For any word w , we denote by $\mathcal{R}(w)$ the number of maximal repetitions in w and by $\mathcal{E}(w)$ the sum of exponents of all maximal repetitions in w . The following statements are proved in [16].

Theorem 1 $\max_{|w|=n} \mathcal{E}(w) = O(n)$.

Corollary 1 $\max_{|w|=n} \mathcal{R}(w) = O(n)$.

A series of papers (e.g., [6, 8]) focused on more precise upper bounds on $\mathcal{E}(w)$ and $\mathcal{R}(w)$ trying to obtain the best possible constant factor behind the O -notation. A breakthrough in this direction was recently made in [2] where the so-called “runs conjecture” $\mathcal{R}(w[1..n]) < n$ was proved. To the best of our knowledge, the currently best upper bound $\mathcal{R}(w[1..n]) \leq \frac{22}{23}n$ on $\mathcal{R}(w)$ is shown in [11].

On the algorithmic side, an $O(n)$ -time algorithm for finding all runs in a word of length n was proposed in [16] for the case of constant-size alphabet. Another $O(n)$ -time algorithm, based on a different approach, has been proposed in [2]. The $O(n)$ time bound holds for the (polynomially-bounded) integer alphabet as well, see, e.g., [2]. However, for the case of unbounded-size alphabet where characters can only be tested for equality, the lower bound $\Omega(n \log n)$ on computing all runs has been known for a long time [23]. It is an interesting open question (raised over 20 years ago in [3]) whether the $O(n)$ bound holds for an unbounded linearly-ordered alphabet. Some results related to this question have recently been obtained in [21].

Gapped repeats and subrepetitions. Another natural generalization of squares are factors of the form uvu where u and v are nonempty words. We call such factors *gapped repeats*. For a gapped repeat uvu , the left (resp. right) occurrence of u is called the *left* (resp. *right*) *copy*, and v is called the *gap*. The *period* of this gapped repeat is $|u| + |v|$. For a gapped repeat π , we denote the length of copies of π by $c(\pi)$ and the period of π by $p(\pi)$. Note that a gapped repeat $\pi = uvu$ may have different periods, and $per(\pi) \leq p(\pi)$. For example, in string `cabacaabaa`, segment `abacaaba` corresponds to two gapped repeats having copies `a` and `aba` and periods 7 and 5 respectively.

Gapped repeats forming the same segment but having different periods are considered distinct. This means that to specify a gapped repeat it is generally not sufficient to specify its segment. If u', u'' are equal non-overlapping factors and u' occurs to the left of u'' , then by (u', u'') we denote the gapped repeat with left copy u' and right copy u'' . For a given gapped repeat (u', u'') , equal factors $u'[i..j]$ and $u''[i..j]$, for $1 \leq i \leq j \leq |u'|$, of the copies u', u'' are called *corresponding factors* of repeat (u', u'') .

For any real $\alpha > 1$, a gapped repeat π is called α -gapped if $p(\pi) \leq \alpha c(\pi)$. Maximality of gapped repeats is defined similarly to repetitions. A gapped repeat $(w[i'..j'], w[i''..j''])$ in w is called *maximal* if it satisfies the following conditions:

1. $w[i' - 1] \neq w[i'' - 1]$ if $i' > 1$,
2. $w[j' + 1] \neq w[j'' + 1]$ if $j'' < n$.

In other words, a gapped repeat π is maximal if its copies cannot be extended to the left nor to the right by at least one letter without breaking its period $p(\pi)$. As observed in [19], any α -gapped repeat is contained either in a (unique) maximal α -gapped repeat with the same period, or in a (unique) maximal repetition with a period which is a divisor of the repeat's period. For example, in the above string `cabacaabaa`, gapped repeat `(ab)aca(ab)` is contained in maximal repeat `(aba)ca(aba)` with the same period 5. In string `cabaaabaaa`, gapped repeat `(ab)aa(ab)` with period 4 is contained in maximal repetition `abaaabaaa` with period 4. Since all maximal repetitions can be computed efficiently in $O(n)$ time (see above), the problem of computing all α -gapped repeats in a word can be reduced to the problem of finding all maximal α -gapped repeats.

Several variants of the problem of computing gapped repeats have been studied earlier. In [4], it was shown that all maximal gapped repeats with a gap length belonging to a specified interval can be found in time $O(n \log n + S)$, where n is the word length and S is output size. In [20], an algorithm was proposed for finding all gapped repeats with a fixed gap length d running in time $O(n \log d + S)$. In [19], it was proved that the number of maximal α -gapped repeats in a word of length n is bounded by $O(\alpha^2 n)$ and all maximal α -gapped repeats can be found in $O(\alpha^2 n)$ time for the case of integer alphabet. A new approach to computing gapped repeats was recently proposed in [13, 10]. In particular, in [13] it is shown that the longest α -gapped repeat in a word of length n over an integer alphabet can be found in $O(\alpha n)$

time. Finally, in a recent paper [25], an algorithm is proposed for finding all maximal α -gapped repeats in $O(\alpha n + S)$ time where S is the output size, for a constant-size alphabet. The algorithm uses an approach previously introduced in [1].

Recall that repetitions are segments with exponent at least 2. Another way to approach gapped repeats is to consider segments with exponent smaller than 2, but strictly greater than 1. Clearly, such a segment corresponds to a gapped repeat $\pi = uvu$ with $per(\pi) = p(\pi) = |u| + |v|$. We will call such factors (segments) *subrepetitions*. More precisely, for any δ , $0 < \delta < 1$, by a δ -subrepetition we mean a factor v that satisfies $1 + \delta \leq exp(v) < 2$. Again, the notion of maximality straightforwardly applies to subrepetitions as well: maximal subrepetitions are defined exactly in the same way as maximal repetitions. The relationship between maximal subrepetitions and maximal gapped repeats was clarified in [19]. Directly from the definitions, a maximal subrepetition π in a string w corresponds to a maximal gapped repeat with $p(\pi) = per(\pi)$. Furthermore, a maximal δ -subrepetition corresponds to a maximal $\frac{1}{\delta}$ -gapped repeat. However, there may be more maximal $\frac{1}{\delta}$ -gapped repeats than maximal δ -subrepetitions, as not every maximal $\frac{1}{\delta}$ -gapped repeat corresponds to a maximal δ -subrepetition.

Some combinatorial results on the number of maximal subrepetitions in a string were obtained in [18]. In particular, it was proved that the number of maximal δ -subrepetitions in a word of length n is bounded by $O(\frac{n}{\delta} \log n)$. In [19], an $O(n/\delta^2)$ bound on the number of maximal δ -subrepetitions in a word of length n was obtained. Moreover, in [19], two algorithms were proposed for finding all maximal δ -subrepetitions in the word running respectively in $O(\frac{n \log \log n}{\delta^2})$ time and in $O(n \log n + \frac{n}{\delta^2} \log \frac{1}{\delta})$ expected time, over the integer alphabet. In [1], it is shown that all subrepetitions with the largest exponent (over all subrepetitions) can be found in an overlap-free string in time $O(n)$, for a constant-size alphabet.

Our results. In the present work we improve the results of [19] on maximal gapped repeats: we prove an asymptotically tight bound of $O(\alpha n)$ on the number of maximal α -gapped repeats in a word of length n (Section 3). From our bound, we also derive a $O(n/\delta)$ bound on the number of maximal δ -subrepetitions occurring in the word, which improves the bound of [18] by a $\log n$ factor. Then, based on the algorithm of [13], we obtain an asymptotically optimal $O(\alpha n)$ time bound for computing all maximal α -gapped repeats

in a string (Section 4). Note that this bound follows from the recently published paper [25] that presents an $O(\alpha n + S)$ algorithm for computing all maximal α -gapped repeats. Here we present an alternative algorithm with the same bound that we obtained independently.

2 Preliminaries

In this Section we state a few propositions that will be used later in the paper. The following fact is well-known (see, e.g., [15, Proposition 2]).

Proposition 1 *Any period p of a word v such that $|v| \geq 2p$ is divisible by $\text{per}(v)$, the smallest period of v .*

Let Δ be some natural number. A period p of some word v is called Δ -period if p is divisible by Δ . The minimal Δ -period of v , if exists, is denoted by $p_\Delta(v)$. The word v is called Δ -periodic if $|v| \geq 2p_\Delta(v)$. It is obvious that any Δ -periodic word is also periodic. Proposition 1 can be generalized in the following way.

Proposition 2 *Any Δ -period p of a word v such that $|v| \geq 2p$ is divisible by $p_\Delta(v)$.*

Proof. By Proposition 1, period p is divisible by $\text{per}(v)$, so p is divisible by $\text{LCM}(\text{per}(v), \Delta)$. On the other hand, $\text{LCM}(\text{per}(v), \Delta)$ is a Δ -period of v . Thus, $p_\Delta(v) = \text{LCM}(\text{per}(v), \Delta)$, and p is divisible by $p_\Delta(v)$. ■

Consider an arbitrary word $w = w[1..n]$ of length n . Recall that any repetition y in w is extended to a unique maximal repetition r with the same minimal period. We call r the *extension* of y .

Let r be a repetition in the word w . We call any factor of w of length $\text{per}(r)$ which is contained in r a *cyclic root* of r . For cyclic roots we have the following property proved, e.g., in [19, Proposition 2].

Proposition 3 *Two cyclic root u' , u'' of a repetition r are equal if and only if $\text{beg}(u') \equiv \text{beg}(u'') \pmod{\text{per}(r)}$.*

3 Number of maximal repeats and subrepetitions

In this section, we obtain an improved upper bound on the number of maximal gapped repeats and subrepetitions in a string w . Following the general approach of [19], we split all maximal gapped repeats into three categories according to periodicity properties of repeat's copy: periodic, semiperiodic and ordinary repeats. Bounds for periodic and semiperiodic repeats are directly borrowed from [19], while for ordinary repeats, we obtain a better bound.

Periodic repeats. We say that a maximal gapped repeat is *periodic* if its copies are periodic strings (i.e. of exponent at least 2). The set of all periodic maximal α -gapped repeats in w is denoted by \mathcal{PP}_α . The following bound on the size of \mathcal{PP}_α was been obtained in [19, Corollary 6].

Lemma 1 $|\mathcal{PP}_k| = O(kn)$ for any natural $k > 1$.

Semiperiodic repeats. A maximal gapped repeat is called *prefix (suffix) semiperiodic* if the copies of this repeat are not periodic, but have a prefix (suffix) which is periodic and its length is at least half of the copy length. A maximal gapped repeat is *semiperiodic* if it is either prefix or suffix semiperiodic. The set of all semiperiodic α -gapped maximal repeats is denoted by \mathcal{SP}_α . In [19, Corollary 8], the following bound was obtained on the number of semiperiodic maximal α -gapped repeats.

Lemma 2 ([19]) $|\mathcal{SP}_k| = O(kn)$ for any natural $k > 1$.

Ordinary repeats. Maximal gapped repeats which are neither periodic nor semiperiodic are called *ordinary*. The set of all ordinary maximal α -gapped repeats in the word w is denoted by \mathcal{OP}_α . In the rest of this section, we prove that the cardinality of \mathcal{OP}_α is $O(\alpha n)$. For simplicity, assume that α is an integer number k .

To estimate the number of ordinary maximal k -gapped repeats, we use the following idea from [15]. We represent a maximal repeat $\pi \equiv (u', u'')$ from \mathcal{OP}_k by a triple (i, j, c) where $i = \text{beg}(u')$, $j = \text{beg}(u'')$ and $c = c(\pi) = |u'| = |u''|$. Such triples will be called *points*. Obviously, π is uniquely defined by values i, j and c , therefore two different repeats from \mathcal{OP}_k can not be represented by the same point.

For any two points (i', j', c') , (i'', j'', c'') we say that point (i', j', c') *covers* point (i'', j'', c'') if $i' \leq i'' \leq i' + c'/6$, $j' \leq j'' \leq j' + c'/6$, $c' \geq c'' \geq \frac{2c'}{3}$. A point is *covered* by a repeat π if this it is covered by the point representing π . By $V[\pi]$ we denote the set of all points covered by a repeat π . We show that any point can not be covered by two different repeats from \mathcal{OP}_k .

Lemma 3 *Two different repeats from \mathcal{OP}_k cannot cover the same point.*

Proof. Let $\pi_1 \equiv (u'_1, u''_1)$, $\pi_2 \equiv (u'_2, u''_2)$ be two different repeats from \mathcal{OP}_k covering the same point (i, j, c) . Denote $c_1 = c(\pi_1)$, $c_2 = c(\pi_2)$, $p_1 = \text{per}(\pi_1)$, $p_2 = \text{per}(\pi_2)$. Without loss of generality we assume $c_1 \geq c_2$. From $c_1 \geq c \geq \frac{2c_1}{3}$, $c_2 \geq c \geq \frac{2c_2}{3}$ we have $c_1 \geq c_2 \geq \frac{2c_1}{3}$, i.e. $c_2 \leq c_1 \leq \frac{3c_2}{2}$. Note that $w[i]$ is contained in both left copies u'_1, u'_2 , i.e. these copies overlap. If $p_1 = p_2$, then repeats π_1 and π_2 must coincide due to the maximality of these repeats. Thus, $p_1 \neq p_2$. Denote $\Delta = |p_1 - p_2| > 0$. From $\text{beg}(u'_1) \leq i \leq \text{beg}(u'_1) + c_1/6$ and $\text{beg}(u''_1) \leq j \leq \text{beg}(u''_1) + c_1/6$ we have

$$(j - i) - c_1/6 \leq p_1 \leq (j - i) + c_1/6.$$

Analogously, we have

$$(j - i) - c_2/6 \leq p_2 \leq (j - i) + c_2/6.$$

Thus $\Delta \leq (c_1 + c_2)/6$ which, together with inequality $c_1 \leq \frac{3c_2}{2}$, implies $\Delta \leq \frac{5c_2}{12}$.

First consider the case when one of the copies u'_1, u'_2 is contained in the other, i.e. u'_2 is contained in u'_1 . In this case, u''_1 contains some factor \hat{u}''_2 corresponding to the factor u'_2 in u'_1 . Since $\text{beg}(u''_2) - \text{beg}(u'_2) = p_2$, $\text{beg}(\hat{u}''_2) - \text{beg}(u'_2) = p_1$ and $u''_2 = \hat{u}''_2 = u'_2$, we have

$$|\text{beg}(u''_2) - \text{beg}(\hat{u}''_2)| = \Delta,$$

so Δ is a period of u''_2 such that $\Delta \leq \frac{5}{12}c_2 = \frac{5}{12}|u''_2|$. Thus, u''_2 is periodic which contradicts that π_2 is not periodic.

Now consider the case when u'_1, u'_2 are not contained in one another. Denote by z' the overlap of u'_1 and u'_2 . Let z' be a suffix of u'_k and a prefix of u'_l where $k, l = 1, 2$, $k \neq l$. Then u''_k contains a suffix z'' corresponding to the suffix z' in u'_k , and u''_l contains a prefix \hat{z}'' corresponding to the prefix z' in u'_l . Since $\text{beg}(z'') - \text{beg}(z') = p_k$ and $\text{beg}(\hat{z}'') - \text{beg}(z') = p_l$ and $z'' = \hat{z}'' = z'$, we have

$$|\text{beg}(z'') - \text{beg}(\hat{z}'')| = |p_k - p_l| = \Delta,$$

therefore Δ is a period of z' . Note that in this case

$$\text{beg}(u'_k) < \text{beg}(u'_l) \leq i \leq \text{beg}(u'_k) + c_k/6,$$

therefore $0 < \text{beg}(u'_l) - \text{beg}(u'_k) \leq c_k/6$. Thus

$$|z'| = c_k - (\text{beg}(u'_l) - \text{beg}(u'_k)) \geq \frac{5}{6}c_k \geq \frac{5}{6}c_2.$$

From $\Delta \leq \frac{5}{12}c_2$ and $c_2 \leq \frac{6}{5}|z'|$ we obtain $\Delta \leq |z'|/2$. Thus, z' is a periodic suffix of u'_k such that $|z'| \geq \frac{5}{6}|u'_k|$, i.e. π_k is either suffix semiperiodic or periodic which contradicts $\pi_k \in \mathcal{OP}_k$. ■

Denote by \mathcal{Q}_k the set of all points (i, j, c) such that $1 \leq i, j, c \leq n$ and $i < j \leq i + (\frac{3}{2}k + \frac{1}{4})c$.

Lemma 4 *Any point covered by a repeat from \mathcal{OP}_k belongs to \mathcal{Q}_k .*

Proof. Let a point (i, j, c) be covered by some repeat $\pi \equiv (u', u'')$ from \mathcal{OP}_k . Denote $c' = c(\pi)$. Note that $w[i]$ and $w[j]$ are contained respectively in u' and u'' and $n > c' \geq c \geq \frac{2c'}{3} > 0$, so inequalities $1 \leq i, j, c \leq n$ and $i < j$ are obvious. Note also that

$$j \leq \text{beg}(u'') + c'/6 = \text{beg}(u') + \text{per}(\pi) + c'/6 \leq i + kc' + c'/6,$$

therefore, taking into account $c' \leq \frac{3c}{2}$, we have $j \leq i + (\frac{3}{2}k + \frac{1}{4})c$. ■

From Lemmas 3 and 4, we obtain

Lemma 5 $|\mathcal{OP}_k| = O(nk)$.

Proof. Assign to each point (i, j, c) the weight $\rho(i, j, c) = 1/c^3$. For any finite set A of points, we define

$$\rho(A) = \sum_{(i,j,c) \in A} \rho(i, j, c) = \sum_{(i,j,c) \in A} \frac{1}{c^3}.$$

Let π be an arbitrary repeat from \mathcal{OP}_k represented by a point (i', j', c') . Then

$$\begin{aligned} \rho(V[\pi]) &= \sum_{i' \leq i \leq i' + c'/6} \sum_{j' \leq j \leq j' + c'/6} \sum_{2c'/3 \leq c \leq c'} \frac{1}{c^3} \\ &> \frac{c'^2}{36} \sum_{2c'/3 \leq c \leq c'} \frac{1}{c^3}. \end{aligned}$$

Using a standard estimation of sums by integrals, one can deduce that $\sum_{2c'/3 \leq c \leq c'} \frac{1}{c^3} \geq \frac{5}{32} \frac{1}{c'^2}$ for any c' . Thus, for any π from \mathcal{OP}_k

$$\rho(V[\pi]) > \frac{1}{36} \frac{5}{36} = \Omega(1).$$

Therefore,

$$\sum_{\pi \in \mathcal{OP}_k} \rho(V[\pi]) = \Omega(|\mathcal{OP}_k|). \quad (1)$$

Note also that

$$\begin{aligned} \rho(\mathcal{Q}_k) &\leq \sum_{i=1}^n \sum_{i < j \leq i + (\frac{3}{2}k + \frac{1}{4})c} \sum_{c=1}^n \frac{1}{c^3} \\ &< n(\frac{3}{2}k + \frac{1}{4})c \sum_{c=1}^n \frac{1}{c^3} < 2nk \sum_{c=1}^n \frac{1}{c^2} < 2nk \sum_{c=1}^{\infty} \frac{1}{c^2} = \frac{nk\pi^2}{3}. \end{aligned}$$

Thus,

$$\rho(\mathcal{Q}_k) = O(nk). \quad (2)$$

By Lemma 4, any point covered by repeats from \mathcal{OP}_k belongs to \mathcal{Q}_k . On the other hand, by Lemma 3, each point of \mathcal{Q}_k can not be covered by two repeats from \mathcal{OP}_k . Therefore,

$$\sum_{\pi \in \mathcal{OP}_k} \rho(V[\pi]) \leq \rho(\mathcal{Q}_k).$$

Thus, using 1 and 2, we conclude that $|\mathcal{OP}_k| = O(nk)$. ■

Putting together Lemma 1, Lemma 2, and Lemma 5, we obtain that for any integer $k \geq 2$, the number of maximal k -gapped repeats in w is $O(nk)$. The bound straightforwardly generalizes to the case of real $\alpha > 1$. Thus, we conclude with

Theorem 2 *For any $\alpha > 1$, the number of maximal α -gapped repeats in w is $O(\alpha n)$.*

Note that the bound of Theorem 2 is asymptotically tight. To see this, it is enough to consider word $w_k = (0110)^k$. It is easy to check that for a big enough α and $k = \Omega(\alpha)$, w_k contains $\Theta(\alpha|w_k|)$ maximal α -gapped repeats whose copies are single-letter words.

We now use Theorem 2 to obtain an upper bound on the number of maximal δ -subrepetitions. The following proposition, shown in [19, Proposition 3], follows from the fact that each maximal δ -subrepetition defines at least one maximal $1/\delta$ -gapped repeat (cf. Introduction).

Proposition 4 ([19]) *For $0 < \delta < 1$, the number of maximal δ -subrepetitions in a string is no more than the number of maximal $1/\delta$ -gapped repeats.*

Theorem 2 combined with Proposition 4 immediately imply the following upper bound for maximal δ -subrepetitions that improves the bound of [18] by a $\log n$ factor.

Theorem 3 *For $0 < \delta < 1$, the number of maximal δ -subrepetitions in w is $O(n/\delta)$.*

The $O(n/\delta)$ bound on the number of maximal δ -subrepetitions is asymptotically tight, at least on an unbounded alphabet : word $\mathbf{ab}_1\mathbf{ab}_2\ldots\mathbf{ab}_k$ contains $\Omega(n/\delta)$ maximal δ -subrepetitions for $\delta \leq 1/2$.

4 Computing all maximal α -gapped repeats

In this section, we present an $O(\alpha n + S)$ algorithm for computing all maximal α -gapped repeats in a word w . This bound has been recently announced in [25], here we present a different solution. Together with the $O(\alpha n)$ bound of Theorem 2, this implies an $O(\alpha n)$ -time algorithm.

4.1 Computing PR-repeats

Some maximal α -gapped repeats can be specifically located as defined below within maximal repetitions (runs). For example, word $\mathbf{cabababababaa}$ contains maximal gapped repeats $(\mathbf{a})\mathbf{babababab}(\mathbf{a})$, $(\mathbf{aba})\mathbf{babab}(\mathbf{aba})$ and $(\mathbf{ababa})\mathbf{b}(\mathbf{ababa})$ within the run $\mathbf{abababababa} = (\mathbf{ab})^{11/2}$. In this section, we describe the structure of such repeats, and in particular those of them which are periodic (see Section 3), like the repeat $(\mathbf{ababa})\mathbf{b}(\mathbf{ababa})$ above. We show how those maximal α -gapped repeats can be extracted from the runs. Repeats which are located within runs but are not periodic will be found separately, together with repeats (periodic or not) which are not located within runs. This part will be described in the next section.

Let $\pi \equiv (u', u'')$ be a periodic gapped repeat. If the extensions of u' and u'' are the same repetition r then we say that r *generates* π and we call π *PR-repeat* (abbreviating from *Periodic Run-generated*). Gapped repeats which are not PR-repeats are called *non-PR repeats*. We will use the following fact.

Proposition 5 *Let $\pi \equiv (u', u'')$ be a maximal gapped repeat such that its copies u' and u'' contain a pair of corresponding factors having the same extension r . Then π is generated by r .*

Proof. Observe that to prove the proposition, it is enough to show that both copies u' and u'' are contained in r , i.e. $\text{beg}(r) \leq \text{beg}(u')$ and $\text{end}(r) \geq \text{end}(u'')$. Let $\text{beg}(r) > \text{beg}(u')$. Then both letters $w[\text{beg}(r) - 1]$ and $w[\text{beg}(r) - 1 + \text{per}(r)]$ are contained in u' . Let these letters be respectively j -th and $(j + \text{per}(r))$ -th letters of u' . Then we have $u''[j] = u'[j] \neq u'[j + \text{per}(r)] = u''[j + \text{per}(r)]$, i.e. $u''[j] \neq u''[j + \text{per}(r)]$, which is a contradiction to the fact that both letters $u''[j]$ and $u''[j + \text{per}(r)]$ are contained in r . Relation $\text{end}(r) \geq \text{end}(u'')$ is proved analogously. ■

All maximal PR-repeats can be easily computed according to the following lemma.

Lemma 6 *A maximal gapped periodic repeat $\pi \equiv (u', u'')$ is generated by a maximal repetition r if and only if $p(\pi)$ is divisible by $\text{per}(r)$ and*

$$\begin{aligned} |r|/2 < p(\pi) &\leq |r| - 2\text{per}(r), \\ u' &\equiv w[\text{beg}(r) \dots \text{end}(r) - p(\pi)], \\ u'' &\equiv w[\text{beg}(r) + \text{per}(r) \dots \text{end}(r)]. \end{aligned}$$

Proof. Let π be generated by r . Consider prefixes of u' and u'' of length $\text{per}(r)$. These prefixes are equal cyclic roots of r , and by Proposition 3 the difference $\text{beg}(u'') - \text{beg}(u') = p(\pi)$ is divisible by $\text{per}(r)$. Inequalities $|r|/2 < p(\pi) \leq |r| - 2\text{per}(r)$ follow immediately from the definition of a repeat generated by a repetition. To prove the last two conditions of the lemma, it is sufficient to prove $\text{beg}(u') = \text{beg}(r)$ and $\text{end}(u'') = \text{end}(r)$. Let $\text{beg}(u') \neq \text{beg}(r)$, i.e. $\text{beg}(u') > \text{beg}(r)$. Then both letters $w[\text{beg}(u') - 1]$ and $w[\text{beg}(u'') - 1]$ are contained in r . Thus, since the difference $(\text{beg}(u'') - 1) - (\text{beg}(u') - 1) = p(\pi)$ is divisible by $\text{per}(r)$, we have $w[\text{beg}(u') - 1] = w[\text{beg}(u'') - 1]$ which contradicts the maximality of π . The relation $\text{end}(u'') = \text{end}(r)$ is proved analogously. Thus, all the conditions of the lemma are proved. On the other hand, if π satisfies all the conditions of the lemma then π is obviously generated by r . ■

Corollary 2 *A maximal repetition r generates no more than $\exp(r)/2$ maximal PR-repeats, and all these repeats can be computed from r in $O(\exp(r))$ time.*

To find all maximal α -gapped PR-repeats in a string w , we first compute all maximal repetitions in w in $O(n)$ time (see Introduction). Then, for each maximal repetition r , we output all maximal α -gapped repeats generated by r . Using Corollary 2, this can be done in $O(\exp(r))$ time. Thus the total time of processing all maximal repetitions is $O(\mathcal{E}(w))$. Since $E(w) = O(n)$ by Theorem 1, all maximal α -gapped PR-repeats in w can be computed in $O(n)$ time.

4.2 Computing non-PR repeats

We now turn to the computation of maximal non-PR α -gapped repeats. Recall that non-PR repeats are those which are either non-periodic, or periodic but not located within a single run. Our goal is to show that all maximal non-PR α -gapped repeats can be found in $O(\alpha n)$ time. Observe that there exists a trivial algorithm for computing all maximal α -gapped repeats in $O(n^2)$ time that proceeds as follows: for each period $p \leq n$, find all maximal α -gapped repeats with period p in $O(n)$ time by consecutively comparing symbols $w[i]$ and $w[i + p]$ for $i = 1, 2, \dots, n - p$.

From the results of [4], it follows that all maximal α -gapped repeats can be found in time $O(n \log n + S)$. This, together with Theorem 2, implies an $O(\alpha n)$ -time algorithm for the case $\alpha \geq \log n$. Therefore, we only have to consider the case $\alpha < \log n$.

(i) Preliminaries

Assume that $\alpha < \log n$. For this case, we proceed with a modification of the algorithm of [13]. We compute all maximal α -gapped non-PR repeats π in w such that $c(\pi) \geq \log n$. To do this, we divide w into *blocks* of $\Delta = (\log n)/4$ consecutive symbols of w . Without loss of generality, we assume that $n = 2^k \Delta$, i.e. w contains exactly 2^k blocks. A word x of length $2^l \Delta$ where $0 \leq l \leq k - 1$ is called a *basic factor* of w if $x = w[i\Delta + 1 \dots (i + 2^l)\Delta]$ for some i . Such an occurrence $w[i\Delta + 1 \dots (i + 2^l)\Delta]$ of x starting at a block frontier will be called *aligned*. A basic factor x of length $2^l \Delta$, where $1 \leq l \leq k - 1$, is called *superbasic* if $x = w[i2^l \Delta + 1 \dots (i + 1)2^l \Delta]$ for some i .

Note that w contains $O(n)$ aligned occurrences of basic factors and $O(\frac{n}{\log n})$ aligned occurrences of superbasic factors. Let $z \equiv w[q2^l\Delta + 1 \dots (q+1)2^l\Delta]$ be an aligned occurrence of superbasic factor of length 2^l in w . For $\tau = 0, 1, \dots, \Delta-1$, an occurrence $w[q2^l\Delta + 1 + \tau \dots (q2^l + 2^{l-1})\Delta + \tau]$ of a basic factor of length $2^{l-1}\Delta$ is called τ -associated (or simply *associated*) with z . Note that any basic factor occurrence τ -associated with z is entirely contained in z and is uniquely defined by z and τ . Thus, z has no more than Δ associated occurrences of basic factors.

To continue, we need one more definition : for $1 \leq i, j \leq n$, denote by $LCP(i, j)$ the length of the longest common prefix of $w[i \dots n]$ and $w[j \dots n]$, and by $LCS(i, j)$ the length of the longest common suffix of $w[1 \dots i]$ and $w[1 \dots j]$.

Let $\pi \equiv (u', u'')$ be a maximal gapped repeat in w such that $c(\pi) \geq \log n = 4\Delta$. Note that in this case, the left copy u' contains at least one aligned occurrence of superbasic factors. Consider aligned occurrences of superbasic factors of maximal length contained in u' . Note that u' can contain either one or two adjacent such occurrences. Let z be the leftmost of them. Note that in this case, we have the following restrictions imposed on u' :

$$\begin{aligned} beg(z) - |z| &< beg(u') \leq beg(z), \\ end(z) &\leq end(u') < end(z) + 2|z|. \end{aligned} \tag{3}$$

Thus, $c(\pi) < 4|z|$. Consider factor z'' in u'' corresponding to z in u' . Note that z'' can be non-aligned. Consider in z'' the leftmost aligned basic factor y'' of length $|z''|/2$. Observe that $beg(z'') \leq beg(y'') < beg(z'') + \Delta$ and y'' is entirely contained in z'' . Let y' be the factor of z corresponding to factor y'' in z'' . It is easily seen that y' is an occurrence of a basic factor associated with z , and π is uniquely defined by z , y' and y'' . Thus, any maximal gapped repeat π such that $c(\pi) \geq \log n$ is uniquely defined by a triple (z, y', y'') , where z is an aligned occurrence of some superbasic factor, y' is an occurrence of some basic factor associated with z , and y'' is an aligned occurrence of the same basic factor. From now on, we will say in such case that π is defined by the triple (z, y', y'') .

Observe that $\pi \equiv (u', u'')$ can be retrieved from (z, y', y'') using LCP and LCS functions.

$$\begin{aligned} beg(u') &= beg(y') - LCS(beg(y') - 1, beg(y'') - 1), \\ end(u') &= end(y') + LCP(end(y') + 1, end(y'') + 1), \\ beg(u'') &= beg(y'') - LCS(beg(y') - 1, beg(y'') - 1), \\ end(u'') &= end(y'') + LCP(end(y') + 1, end(y'') + 1). \end{aligned} \tag{4}$$

Assume additionally that π is an α -gapped repeat for $\alpha > 1$. Then, taking into account inequalities (3) and $c(\pi) < 4|z|$, we have

$$\begin{aligned} \text{end}(y'') &\leq \text{end}(u'') = \text{end}(u') + \text{per}(\pi) < \text{end}(z) + 2|z| + \alpha c(\pi) \\ &< \text{end}(z) + 2|z| + 4\alpha|z| < \text{end}(z) + 6\alpha|z| = \text{end}(z) + 12\alpha|y''|. \end{aligned}$$

On the other hand, $\text{beg}(y'') \geq \text{beg}(u'') > \text{end}(u') \geq \text{end}(z)$. Thus, for any triple (z, y', y'') defining a maximal α -gapped repeat in w the occurrence y'' is contained in the segment $w[\text{end}(z)+1 \dots \text{end}(z)+12\alpha|y'']$ of length $12\alpha|y''|$ to the right of z . We will denote this segment by $\mathcal{I}(z)$. The main idea of the algorithm is to consider all triples (z, y', y'') which can define maximal α -gapped non-PR repeats and for each such triple, check if it actually defines one, which is then computed and output. All the triples (z, y', y'') are considered in a natural way: for each aligned occurrence z of a superbasic factor and each occurrence y' of a basic factor associated with z , we consider all aligned occurrences y'' of the same basic factor in the segment $\mathcal{I}(z)$.

(ii) Naming basic factors on a suffix tree and computing their associated occurrences

We now describe how this computation is implemented. First we construct a suffix tree for the input string w . Suffix tree is a classical data structure of size $O(n)$ which can be constructed in $O(n)$ for a word over constant alphabet see e.g. [14]. Using the suffix tree, we can make in $O(n)$ -time preprocessing which allows to retrieve $LCP(i, j)$ for any i, j in constant time, see e.g. [14]. Similarly, we precompute w to support $LCS(i, j)$ for any i, j in constant time. Then we compute all basic factors of w . This computation is performed by naming all the basic factors, i.e. assigning to each aligned occurrence of a basic factor a name of this factor. The most convenient way to name basic factors is to assign to a basic factor y of length 2^l a pair (l, i) , where i is the start position of the leftmost aligned occurrence of y in w . Note that since we have only n/Δ distinct start positions i , the size of the two-dimensional array required for working with these pairs is $O(n)$. To perform the required computation, we first mark in the suffix tree each node labeled by a basic factor by the name of this factor (in the case when this node is implicit we make it explicit). To this end, for each node v of the suffix tree we compute the value $\text{minleaf}(v)$ which is the smallest leaf number divisible by Δ in the subtree rooted in v if such a number exists. This can be easily done in $O(n)$

time by a bottom-up traversal of the tree. Then, each suffix tree edge (u, v) such that the string depth of u is less than 2^l , the string depth of v is not less than 2^l , and $\text{minleaf}(v)$ is defined is treated in the following way: if the string depth of v is 2^l , node v is marked by name $(l, \text{minleaf}(v))$, otherwise a new node of string depth 2^l is created within edge (u, v) and marked by name $(l, \text{minleaf}(v))$. The obtained tree will be called *marked suffix tree*. Since we have $O(n)$ distinct basic factors, the marked suffix tree contains no more than $O(n)$ additionally inserted nodes. Thus, this tree has $O(n)$ size and is constructed in $O(n)$ time.

To assign to each aligned occurrence $w[i..i+2^l-1]$ of a basic factor the name of this factor, we perform a depth-first top-down traversal of the marked suffix tree. During the traversal we maintain an auxiliary array *basancestor*: at the first visit of a node marked by a name (l, m) we set *basancestor*[l] to m , and at the second visit of this node we reset *basancestor*[l] to undefined. While during the traversal we get to a leaf i divisible by Δ , for each $l = 0, 1, \dots, k-1$ we identify $w[i..i+2^l-1]$ as an occurrence of the basic factor named by $(l, \text{basancestor}[l])$. Note that this traversal is performed in $O(n)$ time.

Then, we compute all occurrences of basic factors associated with aligned occurrences of superbasic factors. This is done again by a depth-first top-down traversal of the marked suffix tree. During the traversal, we maintain the same auxiliary array *basancestor*. Assume that during the traversal we get to a leaf labelled by a position $q2^p\Delta + 1 + \tau$, where q is odd and $0 \leq \tau < \Delta$. Then for each $l = 0, 1, \dots, p-1$ such that *basancestor*[l] is defined, we identify $w[q2^p\Delta + 1 + \tau..(q2^p + 2^l)\Delta + \tau]$ as an occurrence of the basic factor named $(l, \text{basancestor}[l])$, which is τ -associated with the superbasic factor occurrence $w[q2^p\Delta + 1..(q2^p + 2^{l+1})\Delta]$. Observe that this traversal is performed in $O(n)$ time as well.

(iii) Computing lists of aligned occurrences of basic factors

Let y be a Δ -periodic basic factor (cf Introduction). Note that y is also periodic, and then any occurrence of y in w is a repetition. By Proposition 1, the period $\text{per}(y)$ is a divisor of $p_\Delta(y)$. Given the value $p_\Delta(y)$, we can compute in constant time the extension r of any occurrence y' of a Δ -periodic basic factor y as follows:

$$\text{beg}(r) = \text{beg}(y') - \text{LCS}(\text{beg}(y') - 1, \text{beg}(y') + p_\Delta(y) - 1),$$

$$\text{end}(r) = \text{end}(y') + \text{LCP}(\text{beg}(y') + 1, \text{beg}(y'') - p_\Delta(y) + 1).$$

Using Proposition 2, it is easy to show that any set of all aligned occurrences of y having the same extension is a sequence of occurrences, where the difference between start positions of any two consecutive occurrences is equal to $p_\Delta(y)$, i.e. the start positions of all these occurrences form a finite arithmetic progression with common difference $p_\Delta(y)$. We will call these sets *runs of occurrences*. The following fact can be easily proved.

Proposition 6 *Let y', y'' be two consecutive aligned occurrences of a basic factor y in w . Then $|\text{beg}(y') - \text{beg}(y'')| \leq |y|/2$ if and only if y is Δ -periodic, y' and y'' are contained in the same run of occurrences, and, moreover, $|\text{beg}(y') - \text{beg}(y'')| = p_\Delta(y)$.*

At the next step of the algorithm, in order to effectively select appropriate occurrences y'' in the checked triples (z, y', y'') , for each basic factor y we construct a linked list $\text{alignocc}(y)$ of all aligned occurrences of y in the left-to-right order in w . If y is not Δ -periodic, each item of $\text{alignocc}(y)$ consists of only one aligned occurrence of y defined, for example, by its start position (we will call such items *ordinary*). If y is Δ -periodic, each item of $\text{alignocc}(y)$ contains a run of aligned occurrences of y . If a run of aligned occurrences of y consists of only one occurrence, we will consider the item of $\text{alignocc}(y)$ for this run as ordinary, otherwise, if a run of aligned occurrences of y consists of at least two occurrences, the item of $\text{alignocc}(y)$ for this run will be defined, for example, by start positions of leftmost and rightmost occurrences in the run and the value $p_\Delta(y)$ (such item will be called *runitem*). The following fact follows from Proposition 6.

Proposition 7 *Let y', y'' be two consecutive aligned occurrences of a basic factor y in w . Then $|\text{beg}(y') - \text{beg}(y'')| \leq |y|/2$ if and only if y' and y'' are contained in the same runitem of $\text{alignocc}(y)$ and, moreover, $|\text{beg}(y') - \text{beg}(y'')| = p_\Delta(y)$.*

Proposition 7 implies that if two aligned occurrences y', y'' of a basic factor y are contained in distinct items of $\text{alignocc}(y)$ then $|\text{beg}(y') - \text{beg}(y'')| > |y|/2$. Therefore, we have the following consequence from the proposition.

Corollary 3 *Let y be a basic factor of w . Then for any segment v in w , the list $\text{alignocc}(y)$ contains $O(|v|/|y|)$ items having at least one occurrence of y contained in v .*

To construct the lists *alignocc*, for each $i = 1, 2, \dots, n$ and each $l = 0, 1, \dots, k-1$, we insert consecutively the occurrence $y' \equiv w[i \dots i + 2^l - 1]$ of some basic factor y to the appropriate list *alignocc*(y) as follows. Consider the last item in the current list *alignocc*(y). Let it be an ordinary item consisting of an occurrence y'' of y starting at position j . Denote $\delta = i - j$. Consider the following two cases for δ . Let $\delta > |y|/2$. Then, by Proposition 7, y'' and y' are contained in distinct items of *alignocc*(y), and in this case we insert y' to *alignocc*(y) as a new ordinary item. Now let $\delta \leq |y|/2$. In this case, by Proposition 7, y'' and y' are the first two occurrences of the same run of occurrences of y and, moreover, $\delta = p_\Delta(y)$. Let r be the extension of the occurrences of this run. It is easy to see that

$$\text{end}(r) = \text{end}(y') + \text{LCP}(\text{end}(y'') + 1, \text{end}(y') + 1),$$

i.e. $\text{end}(r)$ can be computed in constant time. From the values $\text{beg}(y'')$, $\text{end}(r)$ and $p_\Delta(y)$ we can compute in constant time the start position of the last occurrence of y in the considered run of occurrences and thereby identify completely this run. Thus, in this case we replace the last item of *alignocc*(y) by the identified run of occurrences of y . Now let the last item in *alignocc*(y) be a run of occurrences. Then, if y' is not contained in this run, we insert y' to *alignocc*(y) as a new ordinary item. Thus, each occurrence of a basic factor in w is processed in constant time, and the total time for construction of lists *alignocc* is $O(n)$.

Furthermore, in order to optimize the selection of appropriate occurrences y'' in the checked triples (z, y', y'') , for each pair (z, y') where z is an aligned occurrence of a superbasic factor and y' is an occurrence of some basic factor y associated with z , we compute a pointer *firstocc*(z, y') to the first item in *alignocc*(y) containing at least one occurrence of y to the right of z . For these purposes, we use auxiliary lists *factends*(i) defined for each position i in w . Lists *factends*(i) consist of pairs (z, y') and are constructed at the stage computation of occurrences associated with aligned occurrences of superbasic factors: each time we find a new occurrence y' associated with an aligned occurrence z of a superbasic factor, we insert the pair (z, y') into the list *factends*($\text{end}(z) + 1$). After construction of lists *alignocc*, we compute consecutively for each $i = 1, 2, \dots, n$ pointers *firstocc*(z, y') for all pairs (z, y') from the list *factends*(i). During the computation, we save in each list *alignocc*(y) the last item pointed before (this item is denoted by *lastpnt*(y)). To compute *firstocc*(z, y'), we go through the list *alignocc*(y) from *lastpnt*(y) (or from the beginning of *alignocc*(y) if *lastpnt*(y) does not exist) until we

find the first item containing at least one occurrence of y to the right of the position i . The found item is pointed by $firstocc(z, y')$ and becomes a new item $lastpnt(y)$. Since the total size of lists $alignocc$ and $factends$ is $O(n)$, the total time of computing $firstocc(z, y')$ is also $O(n)$.

(iv) Main step: computing large repeats

At the main stage of the algorithm, in order to process each pair (z, y') , note that all appropriate for (z, y') occurrences y'' contained in $\mathcal{I}(z)$ are located in the fragment of $alignocc(y)$ consisting of all items having at least one occurrence of y contained in $\mathcal{I}(z)$. We will call this fragment *checked fragment*. Thus, we consider all items of the checked fragment by going through this fragment from the first item which can be found in constant time by the value $firstocc(z, y')$. For each considered item, we check triples (z, y', y'') for all occurrences y'' from this item as follows.

Let the considered item be an ordinary item consisting of only one occurrence y'' . Recall that gapped repeat (u', u'') defined by the triple (z, y', y'') can be computed in constant time by formulas (4). Thus, if (u', u'') is an α -gapped repeat satisfying conditions (3), we output it.

Now let the item considered in the checked fragment be a runitem. This implies that basic factor y is Δ -periodic, i.e y is Δ -periodic. Moreover, from the runitem we can derive the value $p_\Delta(y)$. Therefore we can compute in constant time extensions r' and r'' of occurrences y' and y'' respectively. Denote by ρ the run of occurrences contained in the runitem. Recall that our goal is to compute effectively all α -gapped repeats defined by triples (z, y', y'') such that $y'' \in \rho$. Note that, if r' and r'' are the same repetition, then by Proposition 5 all such repeats are PR-repeats, therefore we can assume that r' and r'' are distinct repetitions. Let (u', u'') be an α -gapped repeat defined by a triple (z, y', y'') where $y'' \in \rho$. First, consider the case when u' is not contained in r' , i.e. either $beg(u') < beg(r')$ or $end(u') > end(r')$.

Proposition 8 *If $beg(u') < beg(r')$, then $beg(r') - beg(u') = beg(r'') - beg(u'')$.*

Proof. Define $\gamma' = beg(r') - beg(u')$, $\gamma'' = beg(r'') - beg(u'')$. Let $\gamma' > \gamma''$. Then $u'[\gamma' + per(y)] \neq u'[\gamma'] = u''[\gamma'] = u''[\gamma' + per(y)]$, i.e. we have a contradiction $u'[\gamma' + per(y)] \neq u''[\gamma' + per(y)]$. Similarly, we obtain a contradiction $u'[\gamma'' + per(y)] \neq u''[\gamma'' + per(y)]$ in the case $\gamma' < \gamma''$. ■

The following proposition can be proved analogously.

Proposition 9 *If $\text{end}(u') > \text{end}(r')$, then $\text{end}(u') - \text{end}(r') = \text{end}(u'') - \text{end}(r'')$.*

Define

$$\begin{aligned} s_{\text{left}} &= \text{beg}(y') + (\text{beg}(r'') - \text{beg}(r')), \\ s_{\text{right}} &= \text{beg}(y') + (\text{end}(r'') - \text{end}(r')). \end{aligned}$$

From Propositions 8 and 9, we derive the following fact.

Corollary 4 *If $\text{beg}(u') < \text{beg}(r')$ then $\text{beg}(y'') = s_{\text{left}}$. If $\text{end}(u') > \text{end}(r')$ then $\text{beg}(y'') = s_{\text{right}}$.*

Thus, for computing α -gapped repeats (u', u'') such that u' is not contained in r' , it is enough to consider in ρ only occurrences y''_{left} and y''_{right} with start positions s_{left} and s_{right} respectively, provided that these occurrences exist. We check the occurrences y''_{left} and y''_{right} in the same way as we did for occurrence y'' in the case of ordinary item. Then, it remains to check all occurrences from ρ except for possible occurrences y''_{left} and y''_{right} . Denote by $\rho' = \rho \setminus \{y''_{\text{left}}, y''_{\text{right}}\}$ the set of all such occurrences. Assume that $|r'| \leq |r''|$, i.e. $s_{\text{left}} \leq s_{\text{right}}$ (the case $|r'| > |r''|$ is similar). In order to check all occurrences from ρ' , we consider the following subsets of ρ' separately: subset ρ'_1 of all occurrences y'' such that $\text{beg}(y'') < s_{\text{left}}$, subset ρ'_2 of all occurrences y'' such that $s_{\text{left}} < \text{beg}(y'') < s_{\text{right}}$, and subset ρ'_3 of all occurrences y'' such that $s_{\text{right}} < \text{beg}(y'')$. Note that start positions of all occurrences in each of these subsets form a finite arithmetic progression with common difference $p_{\Delta}(y)$. Thus, we unambiguously denote all occurrences in each of the subsets ρ'_i , $i = 1, 2, 3$, by $y''_0, y''_1, \dots, y''_k$ where y''_0 is the leftmost occurrence in the subset ρ'_i and $\text{beg}(y''_j) = \text{beg}(y''_0) + jp_{\Delta}(y)$ for $j = 1, \dots, k$. Note that values $\text{beg}(y''_0)$ and k for each subset ρ'_i can be computed in constant time.

First, consider an occurrence y''_j from ρ'_1 . Let $\pi \equiv (u', u'')$ be the repeat defined by triple (z, y', y''_j) . Note that

$$\text{per}(\pi) = \text{beg}(y''_j) - \text{beg}(y') = q + jp_{\Delta}(y), \quad (5)$$

where $q = \text{beg}(y''_0) - \text{beg}(y')$. Taking into account that y' and y''_j are contained in maximal repetitions r' and r'' respectively, it is easy to verify that

$$\begin{aligned} \text{LCS}(\text{beg}(y') - 1, \text{beg}(y''_j) - 1) &= \text{beg}(y''_j) - \text{beg}(r''), \\ \text{LCP}(\text{end}(y') + 1, \text{end}(y''_j) + 1) &= \text{end}(r') - \text{end}(y'). \end{aligned}$$

Therefore, $beg(u') = beg(r'') - per(\pi) = q' - jp_\Delta(y)$, where $q' = beg(r'') - q$, and $end(u') = end(r')$. It follows that

$$c(\pi) = |u'| = end(u') - beg(u') + 1 = q'' + jp_\Delta(y),$$

where $q'' = end(r') + 1 - q'$. Recall that for any α -gapped repeat π , we have $c(\pi) < per(\pi) \leq \alpha c(\pi)$. Thus, π is an α -gapped repeat if and only if

$$q'' < q \leq \alpha q'' + (\alpha - 1)jp_\Delta(y). \quad (6)$$

Moreover, u' has to satisfy conditions (3). Thus, the triple (z, y', y_j'') defines an α -gapped repeat if and only if conditions (6) and (3) are verified for j . Note that all these conditions are linear inequalities on j , and then can be resolved in constant time. Thus, we output all α -gapped repeats defined by triples (z, y', y'') such that $y'' \in \rho'_1$ in time $O(1 + S)$, where S is the size of the output.

Now consider an occurrence y_j'' from ρ'_2 . Let $\pi \equiv (u', u'')$ be the repeat defined by the triple (z, y', y_j'') . Note that in this case, $per(\pi)$ also satisfies relation (5). Analogously to the previous case of set ρ'_1 , we obtain that $beg(u') = beg(r')$ and $end(u') = end(r')$, and then $c(\pi) = |r'|$. Therefore, π is an α -gapped repeat if and only if

$$|r'| < q + jp_\Delta(y) \leq \alpha |r'|. \quad (7)$$

Thus, in this case, we output all α -gapped repeats defined by triples (z, y', y_j'') such that j satisfies conditions (7) and (3). Since all these conditions can be resolved for j in constant time, all these repeats can be output in time $O(1 + S)$ where S is the size of output.

Finally, consider an occurrence y_j'' from ρ'_3 . Let $\pi \equiv (u', u'')$ be the repeat defined by triple (z, y', y_j'') . In this case, $per(\pi)$ also satisfies relation (5). Analogously to the case of set ρ'_1 , we obtain that $beg(u') = beg(r')$ and $end(u') = end(r') - per(\pi) = \hat{q}' - jp_\Delta(y)$, where $\hat{q}' = end(r'') - q$, and then

$$c(\pi) = end(u') - beg(u') + 1 = \hat{q}'' - jp_\Delta(y),$$

where $\hat{q}'' = \hat{q}' - beg(r') + 1$. Therefore, π is an α -gapped repeat if and only if

$$\hat{q}'' - jp_\Delta(y) < q + jp_\Delta(y) \leq \alpha(\hat{q}'' - jp_\Delta(y)). \quad (8)$$

Thus, in this case, we output all α -gapped repeats defined by triples (z, y', y_j'') such that j satisfies conditions (8) and (3). Like in the previous cases, this can be done in time $O(1 + S)$, where S is the size of the output.

Putting together all the considered cases, we conclude that all α -gapped repeats defined by triples (z, y', y'') such that $y'' \in \rho$ can be computed in time $O(1 + S)$ where S is the size of output. Thus, in $O(1 + S)$ time we can process each item of the checked fragment. Therefore, since by Corollary 3 the checked fragment has $O(\alpha)$ items, the total time for processing pair (z, y') is $O(\alpha + S)$ where S is the total number of α -gapped repeats defined by triples (z, y', y'') . Since each occurrence z has no more than Δ associated occurrences y' , the total number of processed pairs (z, y') is $O(n)$. Thus the time complexity of the main stage of the algorithm is $O(\alpha n + S)$, where S is the size of the output. Taking into account that $S = O(\alpha n)$ by Theorem 2, we conclude that the time complexity of the main stage is $O(\alpha n)$. Thus, all maximal α -gapped non-PR repeats π in w such that $c(\pi) \geq \log n$ can be computed in $O(\alpha n)$ time.

(v) Computing small repeats

To compute all remaining maximal α -gapped non-PR repeats in w , note that the length of any such repeat π is not greater than

$$(1 + \alpha)c(\pi) < (1 + \log n) \log n < 2 \log^2 n.$$

Thus, setting $\Delta' = \lfloor 2 \log^2 n \rfloor$, any such repeat is contained in at least one of segments $\mathcal{I}'_i \equiv w[i\Delta' + 1 \dots (i + 2)\Delta']$ for $0 \leq i < n/\Delta'$. Therefore, all the remaining α -gapped repeats can be found by searching separately in segments \mathcal{I}'_i . The procedure of searching for repeats in \mathcal{I}'_i is similar to the algorithm described above. If $\alpha \geq \log \log n$, searching for repeats in \mathcal{I}'_i can be done by the algorithm proposed in [4]. The $O(|\mathcal{I}'_i| \log |\mathcal{I}'_i| + S)$ time complexity implied by this algorithm, where by Theorem 2 the output size S is $O(\alpha |\mathcal{I}'_i|)$, can be bounded here by $O(\alpha \Delta')$. Thus, the total time complexity of the search in all segments \mathcal{I}'_i is $O(\alpha n)$. In the case of $\alpha < \log \log n$, we search in each segment \mathcal{I}'_i for all remaining maximal α -gapped non-PR repeats π in w such that $c(\pi) \geq \log |\mathcal{I}'_i|$ in time $O(\alpha \Delta')$, in the same way as we described above for the word w . The total time of the search in all segments \mathcal{I}'_i is $O(\alpha n)$. Then, it remains to compute all maximal α -gapped non-PR repeats π in w such that $c(\pi) < \log |\mathcal{I}'_i| \leq 3 \log \log n$. Note that the length of any such repeat is not greater than

$$(1 + \alpha)3 \log \log n < (1 + \log \log n)3 \log \log n \leq 6 \log^2 \log n.$$

Thus, setting $\Delta'' = \lfloor 6 \log^2 \log n \rfloor$, any such repeat is contained in at least one of the segments $\mathcal{I}_i'' \equiv w[i\Delta'' + 1 \dots (i+2)\Delta'']$ for $0 \leq i < n/\Delta''$. Note that these segments are words of length $2\Delta''$ over an alphabet of size σ , therefore the total number of distinct segments \mathcal{I}_i'' is not greater than $\sigma^{2\Delta''} \leq \sigma^{12 \log^2 \log n}$. In each of the distinct segments \mathcal{I}_i'' , all maximal α -gapped repeats can be found by the trivial algorithm described above in $O(\Delta''^2) = O(\log^4 \log n)$ time. Thus, maximal α -gapped repeats in all distinct segments \mathcal{I}_i'' can be found in $O(\sigma^{12 \log^2 \log n} \log^4 \log n) = o(n)$ time. We conclude that all remaining maximal α -gapped repeats in w can be found in $O(n + S)$ time where S is the total number of maximal α -gapped repeats contained in all segments \mathcal{I}_i'' . According to Theorem 2, this number can be bounded by $O(\alpha n)$, and the time for finding all the remaining maximal α -gapped repeats can be bounded by $O(\alpha n)$ as well. This leads to the final result.

Theorem 4 *For a fixed $\alpha > 1$, all maximal α -gapped repeats in a word of length n over a constant alphabet can be found in $O(\alpha n)$ time.*

Note that since, as mentioned earlier, a word can contain $\Theta(\alpha n)$ maximal α -gapped repeats, the $O(\alpha n)$ time bound stated in Theorem 4 is asymptotically optimal.

5 Conclusions

Besides gapped repeats we can also consider gapped palindromes which are factors of the form uvu^R where u and v are nonempty words and u^R is the reversal of u [17]. A gapped palindrome uvu^R in a word w is called *maximal* if $w[\text{end}(u) + 1] \neq w[\text{beg}(u^R) - 1]$ and $w[\text{beg}(u) - 1] \neq w[\text{end}(u^R) + 1]$ for $\text{beg}(u) > 1$ and $\text{end}(u^R) < |w|$. A maximal gapped palindrome uvu^R is α -gapped if $|u| + |v| \leq \alpha|u|$ [13]. It can be shown analogously to the results of this paper that for $\alpha > 1$ the number of maximal α -gapped palindromes in a word of length n is bounded by $O(\alpha n)$ and for the case of constant alphabet, all these palindromes can be found in $O(\alpha n)$ time¹.

In this paper we consider maximal α -gapped repeats with $\alpha > 1$. However this notion can be formally generalized to the case of $\alpha \leq 1$. In particular, maximal 1-gapped repeats are maximal repeats whose copies are adjacent

¹Note that in [13], the number of maximal α -gapped palindromes was conjectured to be $O(\alpha^2 n)$.

or overlapping. It is easy to see that such repeats form runs whose minimal periods are divisors of the periods of these repeats. Moreover, each run in a word is formed by at least one maximal 1-gapped repeat, therefore the number of runs in a word is not greater than the number of maximal 1-gapped repeats. More precisely, each run r is formed by $\lfloor \exp(r)/2 \rfloor$ distinct maximal 1-gapped repeats. Thus, if a word contains runs with exponent greater than or equal to 4 then the number of maximal 1-gapped repeats is strictly greater than the number of runs. However, using an easy modification of the proof of “runs conjecture” from [2], it can be also proved the number of maximal 1-gapped repeats in a word is strictly less than the length of the word. Moreover, denoting by $\mathcal{R}(n)$ (respectively, $\mathcal{R}_1(n)$) the maximal possible number of runs (respectively, maximal possible number of maximal 1-gapped repeats) in words of length n , we conjecture that $\mathcal{R}(n) = \mathcal{R}_1(n)$ since known words with a relatively large number of runs have no runs with big exponents. We can also consider the case of $\alpha < 1$ for repeats with overlapping copies, in particular, the case of maximal $1/k$ -gapped repeats where k is integer greater than 1. It is easy to see that such repeats form runs with exponents greater than or equal to $k + 1$. It is known from [2, Theorem 11] that the number of such runs in a word of length n is less than n/k , and it seems to be possible to modify the proof of this fact for proving that the number of maximal $1/k$ -gapped repeats in the word is also less than $n/k = \alpha n$. These observations together with results of computer experiments for the case of $\alpha > 1$ leads to a conjecture that for any $\alpha > 0$, the number maximal α -gapped repeats in a word of length n is actually less than αn . This generalization of the “runs conjecture” constitutes an interesting open problem. Another interesting open question is whether the obtained $O(n/\delta)$ bound on the number of maximal δ -subrepetitions is asymptotically tight for the case of constant alphabet.

Acknowledgments. This work was partially supported by Russian Foundation for Fundamental Research (Grant 15-07-03102).

References

- [1] G. Badkobeh, M. Crochemore, and C. Toopsuwan. Computing the maximal-exponent repeats of an overlap-free string in linear time. In L. Calderón-Benavides, C. N. González-Caro, E. Chávez, and N. Zi-

- viani, editors, *String Processing and Information Retrieval - 19th International Symposium, SPIRE 2012, Cartagena de Indias, Colombia, October 21-25, 2012. Proceedings*, volume 7608 of *Lecture Notes in Computer Science*, pages 61–72. Springer, 2012.
- [2] H. Bannai, T. I. S. Inenaga, Y. Nakashima, M. Takeda, and K. Tsuruta. A new characterization of maximal repetitions by lyndon trees. *CoRR*, abs/1406.0263, 2014.
 - [3] D. Breslauer. *Efficient string algorithmics*. PhD thesis, Columbia University, 1992.
 - [4] G. S. Brodal, R. B. Lyngs, C. N. S. Pedersen, and J. Stoye. Finding maximal pairs with bounded gap. *J. Discrete Algorithms*, 1(1):77–104, 2000.
 - [5] M. Crochemore. An optimal algorithm for computing the repetitions in a word. *Inf. Process. Lett.*, 12(5):244–250, 1981.
 - [6] M. Crochemore, L. Ilie, and L. Tinta. Towards a solution to the ”runs” conjecture. In P. Ferragina and G. M. Landau, editors, *Combinatorial Pattern Matching, 19th Annual Symposium, CPM 2008, Pisa, Italy, June 18-20, 2008, Proceedings*, volume 5029 of *Lecture Notes in Computer Science*, pages 290–302. Springer, 2008.
 - [7] M. Crochemore, C. S. Iliopoulos, M. Kubica, J. Radoszewski, W. Rytter, and T. Walen. Extracting powers and periods in a string from its runs structure. In E. Chávez and S. Lonardi, editors, *String Processing and Information Retrieval - 17th International Symposium, SPIRE 2010, Los Cabos, Mexico, October 11-13, 2010. Proceedings*, volume 6393 of *Lecture Notes in Computer Science*, pages 258–269. Springer, 2010.
 - [8] M. Crochemore, M. Kubica, J. Radoszewski, W. Rytter, and T. Walen. On the maximal sum of exponents of runs in a string. *J. Discrete Algorithms*, 14:29–36, 2012.
 - [9] M. Crochemore and W. Rytter. Squares, cubes, and time-space efficient string searching. *Algorithmica*, 13(5):405–425, 1995.

- [10] M. Dumitran and F. Manea. Longest gapped repeats and palindromes. In G. F. Italiano, G. Pighizzini, and D. Sannella, editors, *Mathematical Foundations of Computer Science 2015 - 40th International Symposium, MFCS 2015, Milan, Italy, August 24-28, 2015, Proceedings, Part I*, volume 9234 of *Lecture Notes in Computer Science*, pages 205–217. Springer, 2015.
- [11] J. Fischer, S. Holub, T. I., and M. Lewenstein. Beyond the runs theorem. *CoRR*, abs/1502.04644, 2015.
- [12] Z. Galil and J. I. Seiferas. Time-space-optimal string matching. *J. Comput. Syst. Sci.*, 26(3):280–294, 1983.
- [13] P. Gawrychowski and F. Manea. Longest α -gapped repeat and palindrome. In A. Kosowski and I. Walukiewicz, editors, *Fundamentals of Computation Theory - 20th International Symposium, FCT 2015, Gdańsk, Poland, August 17-19, 2015, Proceedings*, volume 9210 of *Lecture Notes in Computer Science*, pages 27–40. Springer, 2015.
- [14] D. Gusfield. *Algorithms on Strings, Trees, and Sequences - Computer Science and Computational Biology*. Cambridge University Press, 1997.
- [15] R. Kolpakov. On primary and secondary repetitions in words. *Theor. Comput. Sci.*, 418:71–81, 2012.
- [16] R. Kolpakov and G. Kucherov. On maximal repetitions in words. *J. Discrete Algorithms*, 1(1):159–186, 2000.
- [17] R. Kolpakov and G. Kucherov. Searching for gapped palindromes. *Theor. Comput. Sci.*, 410(51):5365–5373, 2009.
- [18] R. Kolpakov, G. Kucherov, and P. Ochem. On maximal repetitions of arbitrary exponent. *Inf. Process. Lett.*, 110(7):252–256, 2010.
- [19] R. Kolpakov, M. Podolskiy, M. Posypkin, and N. Khrapov. Searching of gapped repeats and subrepetitions in a word. *CoRR*, abs/1309.4055, 2013.
- [20] R. M. Kolpakov and G. Kucherov. Finding repeats with fixed gap. In *SPIRE*, pages 162–168, 2000.

- [21] D. Kosolobov. Lempel-Ziv factorization may be harder than computing all runs. In E. W. Mayr and N. Ollinger, editors, *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, volume 30 of *LIPIcs*, pages 582–593. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.
- [22] M. Lothaire. *Combinatorics on Words*. Addison Wesley, 1983.
- [23] M. G. Main and R. J. Lorentz. Linear time recognition of squarefree strings. *Combinatorial Algorithms on Words*, 1985.
- [24] J. A. Storer. *Data Compression: Methods and Theory*. Computer Science Press, 1988.
- [25] Y. Tanimura, Y. Fujishige, T. I, S. Inenaga, H. Bannai, and M. Takeda. A faster algorithm for computing maximal α -gapped repeats in a string. In C. S. Iliopoulos, S. J. Puglisi, and E. Yilmaz, editors, *String Processing and Information Retrieval - 22nd International Symposium, SPIRE 2015, London, UK, September 1-4, 2015, Proceedings*, volume 9309 of *Lecture Notes in Computer Science*, pages 124–136. Springer, 2015.